A second year progress report on:

Development of a Dynamically Configurable, Object-Oriented Framework for Distributed, Multi-modal Computational Aerospace Systems Simulation

By

Abdollah A. Afjeh, Ph.D.
John A. Reed, Ph.D.

Department of Mechanical, Industrial and Manufacturing Engineering
The University of Toledo

September 7, 2001

Summary

This report describes the progress made in the second year (Sept. 1, 2000 to Aug. 31, 2001) of work at The University of Toledo under the NASA Information Technology (IT) Program grant number NAG-1-2244. This research is aimed at developing a new and advanced simulation framework that will significantly improve the overall efficiency of aerospace systems design and development. This objective will be accomplished through an innovative integration of object-oriented and Web-based technologies with both new and proven simulation methodologies. The basic approach involves three major areas of research:

- Aerospace system and component representation using a hierarchical object-oriented component model which enables the use of multimodels and enforces component interoperability.

- Collaborative software environment that streamlines the process of developing, sharing and integrating aerospace design and analysis models.

- Development of a distributed infrastructure which enables Web-based exchange of models to simplify the collaborative design process, and to support computationally intensive aerospace design and analysis processes.

Research for the second year focused on enabling models developed in the *Denali* software environment to directly access CAD native geometry. Access to CAD geometry is essential to generate mesh for use in fluid and structural analysis of aerospace systems, as well as visualization of analysis results. Furthermore, a *geometry-centric* modeling approach, as employed in this work, simplifies use of these and other tools in a multidisciplinary design process. Finally, *direct* access to CAD native geometry, compared to geometry described in intermediate forms (e.g., IGES[1], STEP[2], STL[3], etc.), is more robust.

Year 2 Accomplishments

Introduction

Computational simulation plays an essential role in the aerospace design process. Computer-aided design (CAD) methods are the basic tool for definition and control of the configuration, and CAD solid modeling capabilities enable designers to create virtual mockups of system to verify that no interferences exist in part layouts. Similarly, structural analysis is almost entirely performed using computational tools employing finite element methods. Computational simulation is also employed to model fluid dynamics. However, computation fluid dynamic (CFD) tools are not as widely applied in the design process as either CAD or structural analysis tools due, in part, to the long set-up times and high costs (both human and computational) associated with complex fluid flow.[4]

The conventional steps for CFD, structural analysis, and other disciplines in the design process are: 1) surface generation, 2) mesh generation, 3) obtaining a solution, and 4) post-processing visualization. Surfaces of the domain to be analyzed (e.g., a turbine blade passage) are generated from a CAD system. These surfaces are used to create a domain (i.e., a closed volume) of interest which is discretized in one of many different manners to form a mesh. The mesh, along with boundary information, is used by a numerical solver to obtain a solution to the governing equations over the entire volume. This solution and mesh are then displayed graphically, allowing the user to examine the results and extract the data needed to understand the domain physics. This process is illustrated in Fig. 1. Data are transmitted between these steps via files; for example, output from a CAD system might be in the form of IGES file(s), which are read by the mesh generator. Similarly, the mesh generator, solvers and visualization tools would each generate output and read input in a variety of formats.

Mesh generation has long been recognized as a bottleneck in the CFD process.[5] While much research on automating the volume mesh generation process have been relatively successful, these methods rely on appropriate initial surface triangulation to work properly. Surface discretization has been one of the least automated steps in computational simulation due to its dependence on implicitly defined CAD surfaces and curves. Differences in CAD geometry engines manifest themselves in discrepancies in their interpretation of the same entities. This lack of "good" geometry causes significant problems for mesh generators, requiring users to "repair" the CAD geometry before mesh generation. The problem is exacerbated when CAD geometry is translated to other forms (e.g., IGES) which do not include important topological and construction information in addition to entity geometry.[6]

One technique to avoid these problems is to access the CAD geometry directly from the mesh generating software, rather than through files. By accessing the geometry model (not a discretized version) in its native environment, this approach avoids translation to a format which can deplete the model of topological information.[6]

Our approach to enable models developed in the Denali software environment to directly access CAD geometry and functions is through an Application Programming Interface (API) known as CAPRI.[7] CAPRI provides a layer of indirection through which CAD-specific data may be accessed by an application program using CAD-system neutral C and FORTRAN language function calls. CAPRI supports a general set of CAD operations such as truth testing, geometry construction and entity queries.

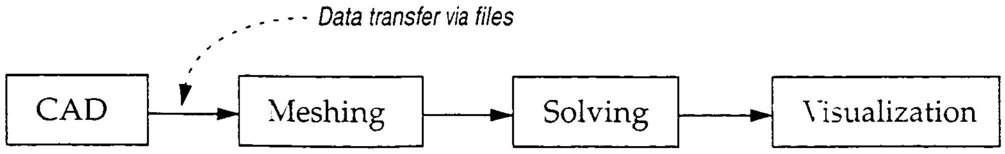

Figure 1: Conventional Analysis Process (Ref. [7])

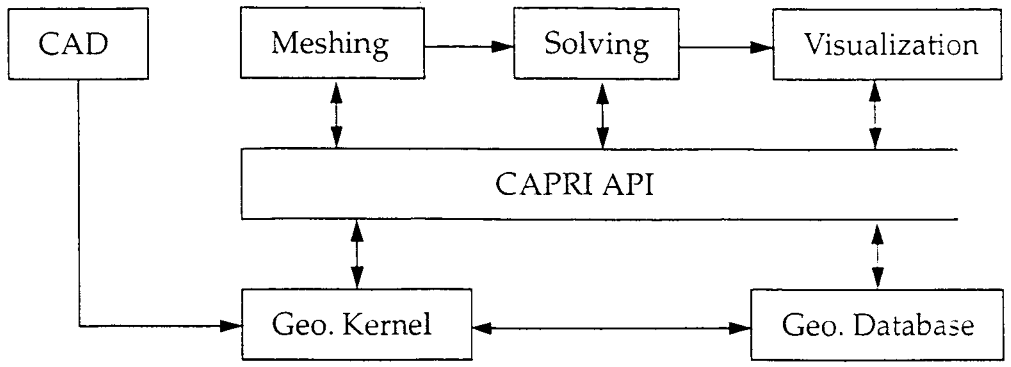

Figure 2: CAPRI-based Analysis Process (Ref. [7])

CAPRI isolates the top level applications (mesh generators, solvers, and visualization programs) from the geometry engine (see Fig. 2). It also allows the replacement of one geometry kernel with another, without affecting the top-level application. Additionally, CAPRI allows non-geometry information, such as material or condition information (e.g., temperature) to be attached to the geometry entities.

A geometry-centric approach, such as the one supported by CAPRI, is vital to foster concurrent engineering, especially in multidisciplinary aerospace design. This approach allows requisite information, both geometric and non-geometric, to be captured and used in the design process. For example, a CFD solver, using the supplied mesh, would generate a solution consisting of fluid properties (e.g., temperature, pressure, etc.) for each volume. This data is attached to appropriate mesh volumes through CAPRI, and accessed by other applications through context-specific *views* of the CAPRI data. For example, a CFD visualization application program would obtain the geometry directly (through CAPRI) from the CAD geometry kernel while the CFD data would be supplied from CAPRI attachments.

Implementation Details: Overview

We have designed and implemented a basic object-oriented architecture to allow both Denali models and external application programs to access geometry data through the CAPRI API. Figure 3 illustrates a simplified view of the architecture participants. The designer directs the Client, which is either a Denali model or external application program, to generate a mesh for a specific CAD part. The MeshGenerator is responsible for generating an appropriate mesh given a CAD part, and is done in conjunction with the CAPRI middleware and a CAD geometry kernel (such as UniGraphics Parasolid). The generated mesh is returned to the Client and passed to the Analysis Controller (it may also be viewed at this time by a visualization tool). The Analysis Controller uses the mesh to perform an engineering analysis, such as CFD. At the end of each time-step or the end of the analysis, CFD data is attached to geometry via calls to CAPRI. The mesh, attached CFD information, and geometry boundary surfaces data are retrieved by a Visualizer which displays the simulation results to the designer.

Mesh Generation

A general class structure has been developed to frame the mesh generation process using CAPRI (see Fig. 4). The MeshGeneratorMgr class provides a single access point (implemented as a Singleton object) for clients to obtain a mesh from a CAD part. There are many different techniques for generating a mesh, so Denali allows users to specify a particular mesh generation technique as implemented by a Java class. These different classes can be dynamically plugged into the Denali framework so long as they subclass the abstract MeshGenerator class. In Fig. 4, the MeshGenerator class has been subclassed by the DenaliMeshGenerator class, which defines concrete implementations of MeshGenerator abstract methods (indicated by italics). MeshGenerator subclass' can use whatever means they wish to generate a mesh; this allows the use of existing IGES- and STEP-based tools. In our research we have written a simple Java mesh

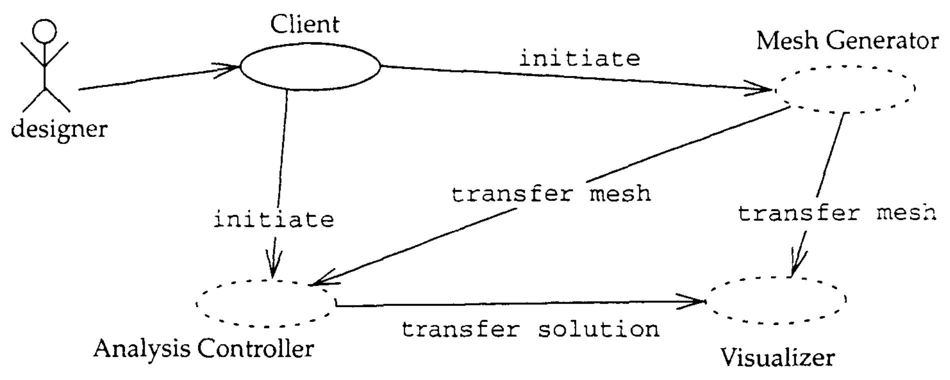

Figure 3: Global view of architecture

generator based on constrained Delaunay triangulation. The generator, which is implemented in the DenaliMeshGenerator class, utilizes CAPRI to access native CAD geometry and generate a mesh. The Capri class is a Java wrapper which duplicates the CAPRI API function call list and accesses the CAPRI C-language function calls through the Java Native Interface (JNI).

Using CAPRI, the DenaiMeshGenerator loads a CAD part, then retrieves a list of volumes from CAPRI. For each volume, CAPRI returns a simplical decomposition of each of the CAD face entities. Each of these triangulations are manifold with respect to their CAD edges. Typically, the triangulation is irregular and planar regions are decomposed into as few triangles as possible. A new mesh with higher quality is constructed by creating additional triangles using points on CAD faces obtained from CAPRI.

Since it was not our intent to write a robust and guaranteed-quality mesh generation tool, we developed the Delaunay triangulation mesh generator only to the point to demonstrate access to geometry through CAPRI. In the future, we may choose to continue this work and improve upon it using the work of Ruppert[9], Chew[10] and Aftosmis.[11].

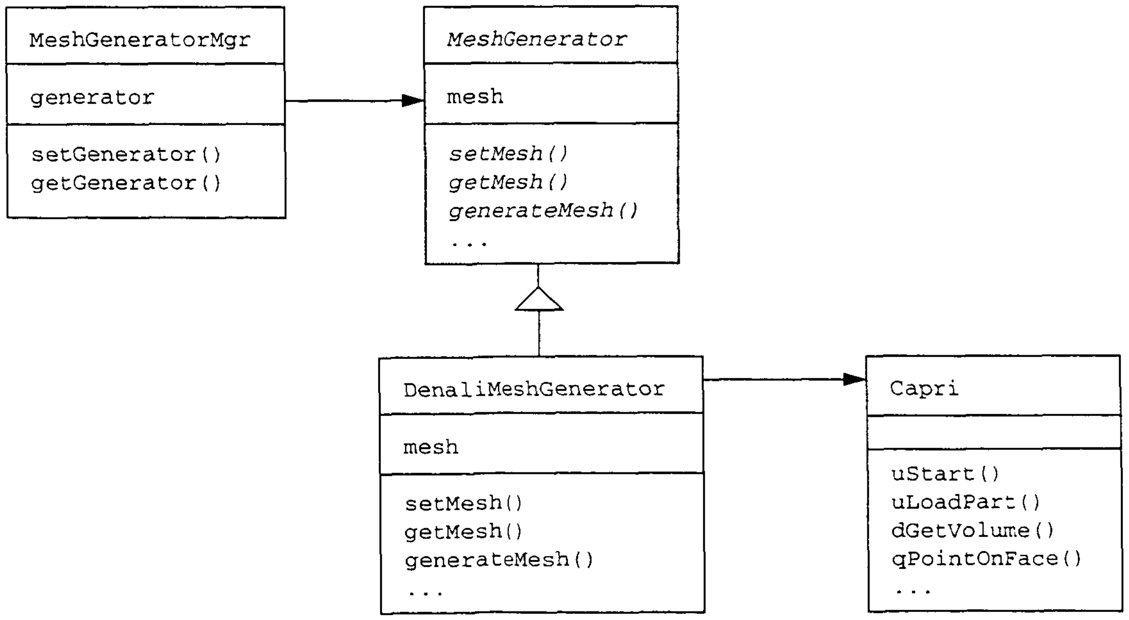

Figure 4: Mesh generation class structure

Visualization

As indicated above, visualization tools are essential to view solver solutions overlaid on geometry and mesh data. One visualization tool, called the *Geometry Viewer,* is a stand-alone visual interface and debugging aid provided with CAPRI. It is similar to the *Visual3* program[12] used for scientific visualization, but is limited to viewing meshes and geometry. We have loosely integrated the Geometry Viewer within the Denali framework so as to demonstrate the ability to visualize geometry and mesh using the CAPRI library.

One of the goals of the Denali framework was to provide a platform-independent system for aerospace design. Towards that end we have endeavored to use Java™ as much as possible in developing the framework. However, in some cases, no Java-based tool were available; this is currently the case with visualization tools. It is sometimes possible to partition the non-Java software into a client-server architecture with the non-Java software located on a centralized machine made accessible via RMI or CORBA. However, it appears that this is not currently possible with existing visualization tools. Consequently we are exploring the possibility of developing a visualization tool similar to *Visual3* or the *Geometry Viewer* using Java, and in particular, the Java3D API.[13] Alternatively, we will have to require users to install platform-specific visualization tools on each desktop using Denali in order to view geometry and/or simulation solutions.

Plans for Year 3

- The majority of work in year 3 will focus on the development of aircraft models for use in Denali. In anticipation of the year 3 work, we have licensed the Base of Aircraft Data (BADA) from the Eurocontrol Experimental Centre (EEC). The Base of Aircraft Data (BADA) provides a set of ASCII files containing performance and operating procedure coefficients for 186 different aircraft types. The coefficients include those used to calculate thrust, drag and fuel flow and those used to specify nominal cruise, climb and descent speeds.
- We will continue to work on implementing a database management system based on the Java Data Objects (JDO) specification.[8] The final JDO specification is expected to be released soon, and we will be evaluating different implementations of the specification to see which is best for supporting Denali.
- We will also be working on integrating more robust grid generator and visualization tools which utilize the CAPRI interface.

References

[1] REED, K., 1991, "The Initial Graphics Exchange Specification (IGES) Version 5.1."

[2] STEP, 1994, "Industrial automation systems and integration — Product data representation and exchange -- Part 1: Overview and fundamental principles," ISO/TR 10303-1. International Standards Org. Geneve, Switzerland.

[3] STL, 1988, *Stereolithography Interface Format Specification*, 3D Systems, Inc.

[4] JAMESON, A., 1999, "Reengineering the Design Process Through Computation," *J. Aircraft*, vol. 36, pp. 36-50.

[5] COSNER, R., 1994, "Issues in aerospace application of CFD analysis," AIAA Paper No. 94-0464.

[6] AFTOSMIS, M.J., DELANAYAE, M., AND HAIMES, R., 1999, "Automatic Generation of CFD-Ready Surface Triangulations from CAD Geometry," AIAA Paper No. 99-0776.

[7] HAIMES, R. AND FOLLEN, G., 1998, "Computational Analysis PRogramming Interface," *Proc. of the 6th International Conference on Numerical Grid Generation in Computational Simulation Fields*, Eds. Cross, Eiseman, Hauser, Soni and Thompson.

[8] JAVA DATA OBJECTS, "JSR 12: Java Data Objects (JDO) Specification," http://jcp.org/jsr/detail/012.jsp

[9] RUPPERT, J. 1995, "A Delaunay Refinement Algorithm for Quality 2-Dimensional Mesh Generation," *J. Algorithms*, vol. 18, no. 3, pp. 548-585.

[10] CHEW, L. P., 1993, "Guaranteed-quality mesh generation for curved surfaces," Proc. of the Ninth Annual Symposium on Computational Geometry, pp. 274-280, ACM.

[11] AFTOSMIS, M.J., 1999, "On the Use of CAD-Native Predicates and Geometry in Surface Meshing," NASA TM-1999-208782.

[12] HAIMES, R., 1991, "Visual3: Interactive Unsteady Unstructured 3D Visualization," AIAA Paper No. 91-0794.

[13] SOWIZRAL, H., RUSHFORTH, K., AND DEERING, M., 2000, *The Java 3DTM API Specification, Second Edition,* Addison Wesley Longman, Inc. ISBN: 0-201-71041-2